I Claim:

1. A method of determining, in a computer environment, the equivalence, if any, of two algebraic expressions for use in compiler optimisation of source code and like computing tasks, said method comprising the steps of:

(a) recasting said expressions into a form of one or more token pairs arranged sequentially in a string, each said token pair comprising an operator followed by an operand;

(b) reducing said strings in accordance with a set of predetermined simplifying rules; and

(c) comparing the reduced strings by matching, to detect equivalence of the two algebraic expressions.

2. A method according to claim 1, whereby the recasting step (a) is preceded by a preconditioning step comprising, in relation to said algebraic expressions, the following sub-steps according to whether a sub step applies:

(da) deleting a space in the expression;

(db) removing a bracket in the expression by expanding a bracketed sub-expression;

(dc) inserting a unitary operator at the start of the expression;

(dd) recasting a power factor, being a variable being raised to a power in the expression, in an alternate form as one of:

(dda) the power factor being expressed as the variable multiplied by itself as many times as the power, if the power is a positive integer;

(ddb) the power factor being expressed as a reciprocal of the variable multiplied by itself as many times as an absolute value of the power, if the power is a negative integer;

(ddc) the power factor being replaced by an appropriate function which can compute the power factor, if the power is not an integer;

(de) recasting a constant in the expression in exponential format;

(df) substituting a "+" operator in the expression by "+1*", a "1" being in exponential format;

(dg) substituting a "-" operator in the expression by "-1*", a "1" being in exponential format; and

(dh) recasting a "division by a constant" in the expression as multiplication by a reciprocal of the constant.

3. A method according to claim 1, whereby the simplifying rules in step (b) comprise:

(ba) arranging token pairs into subgroups;

(bb) arranging operand tokens in an arranged subgroup in order;

(bc) reducing the ordered operands by consolidating one or more constants and eliminating variables of opposite effect to form reduced subgroups; and

(bd) consolidating one or more multiple instances of similar subgroups, to produce a reduced string.

4. A method according to claim 2, whereby an algebraic expression whose equivalence is to be determined contains an aliased variable, said method comprising an additional sub-step of :

arranging an ordered list of aliases of the variable, and substituting a first alias in the ordered list for all instances of the aliased variable in the expression.

5. A method according to claim 2, whereby an algebraic expression whose equivalence is to be determined contains a function, said method comprising additional sub-steps of :

reducing function arguments using the set of predetermined simplifying rules; and

replacing the function by a tagged string, said string designating a function name, parameter types, and arguments, whereby the tag distinguishes the function name from a variable.

6. An apparatus adapted to determine, in a computer environment, the equivalence, if any, of two algebraic expressions for use in compiler optimisation of source code and like computing tasks, said apparatus comprising:

(a) recasting means for recasting said expressions into a form of one or more token pairs arranged sequentially in a string, each said token pair comprising an operator followed by an operand;

(b) reduction means for reducing said strings in accordance with a set of predetermined simplifying rules; and

(c) comparison means for comparing the reduced strings by matching, to detect equivalence of the two algebraic expressions.

7. A computer program product including a computer readable medium having recorded thereon a computer program for determining, in a computer environment,

the equivalence, if any, of two algebraic expressions for use in compiler optimisation of source code and like computing tasks, said program comprising:

(a) recasting process steps for recasting said expressions into a form of one or more token pairs arranged sequentially in a string, each said token pair comprising an operator followed by an operand;

(b) reduction process steps for reducing said strings in accordance with a set of predetermined simplifying rules; and

(c) comparison process steps for comparing the reduced strings by matching, to detect equivalence of the two algebraic expressions.